TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN
MÔN: CÔNG NGHỆ JAVA

ĐỀ TÀI

XÂY DỰNG GAME XẾP HÌNH(TETRIS)

Giảng viên hướng dẫn : Vũ Huấn

Nhóm thực hiện : 1

Thành viên thực hiện : Nguyễn Cảnh Đức 211210351
Nguyễn Đức Thắng 211213487

Lớp : Công nghệ thông tin 5

Khóa : 62

Hà Nội – 2023

I. LỜI MỞ ĐẦU

Công nghệ thông tin(CNTT) ngày càng có vai trò quan trọng trong cuộc sống hằng ngày của chúng ta. Việc ứng dụng CNTT vào các lĩnh vực trong đời sống giúp công việc được tiến hành nhanh chóng và hiệu quả hơn. Có rất nhiều công việc mới phát triển song song với sự phát triển của CNTT, một trong những số đó là hiệu ứng game, hướng đi dịch vụ mang lại hiệu quả kinh tế rất lớn

Chúng em chọn đề tài:"Lập trình game xếp hình bằng ngôn ngữ java" nhằm tìm hiểu sâu hơn về ngôn ngữ java , từ đó viết một ứng dụng cụ thể thử nghiệm làm cơ sở củng cố kiến thức và định hướng, kế hoạch xây dựng những ứng dụng game cụ thể, phát triển theo hướng dịch vụ trong tương lai

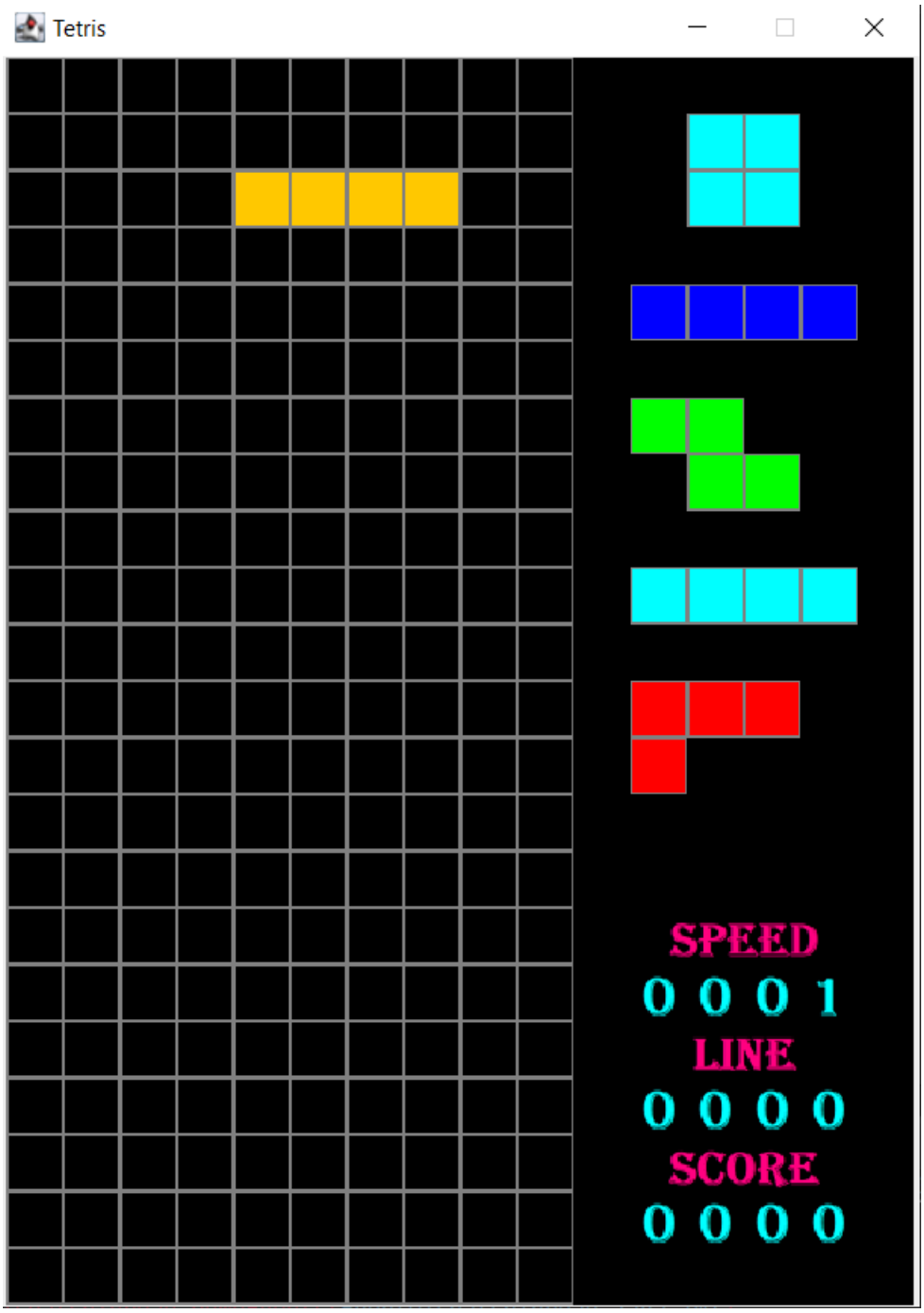
II. PHÂN CÔNG CÔNG VIỆC:

Nguyễn Cảnh Đức	Nguyễn Đức Thắng
Viết giao diện và các logic trong trò chơi Thuyết trình Xây dựng hướng phát triển	Viết chương trình tạo hình và khối Viết báo cáo Xây dựng hướng phát triển

III. HƯỚNG DẪN APP:

1.Mô tả chức năng của game:

a. Giao diện chính



b. Chức năng của game

- Hiển thị khối hình

Chức năng này giúp người chơi biết trước được khối hình nào sẽ xuất hiện trong thao tác kế tiếp, giúp người chơi định hướng dễ dàng hơn.

- Di chuyển và đảo khối
    - Chức năng này cho phép người chơi dịch chuyển và đảo khối theo chiều, hướng và dạng mà mình mong muốn.
        - Dịch chuyển sang trái: Nhấn phím ←
        - Dịch chuyển sang phải: Nhấn Phím →
        - Dịch nhanh xuống dưới: Nhấn phím ↓
        - Đảo gạch: Nhấn phím ↑

- Tính điểm
    - Chức năng này tính điểm cho người chơi, khi người chơi xếp một hàng đầy trên khung, hàng này sẽ tự biến mất và người dùng sẽ được cộng thêm điểm tùy vào số hàng người chơi phá được trong 1 lần phá.
        - Phá 1 hàng 1 lần được cộng thêm 1 điểm
        - Phá 2 hàng 1 lần được cộng thêm 4 điểm
        - Phá 3 hàng 1 lần được cộng thêm 7 điểm
        - Phá 4 hàng 1 lần được cộng thêm 10 điểm

- Tăng độ khó
    - Chức năng này sẽ điều chỉnh độ khó của trò chơi
    - Điều chỉnh theo cơ chế là sẽ tăng độ khó sau khi người chơi phá được 1 số lượng hàng nhất định(trong trò chơi này bọn em cài đặt là cứ sau 10 hàng thì sẽ tăng độ khó lên)

- Tăng độ khó là tăng tốc độ rơi của hình khối

2 Code

a. Code hình vuông

```
package Class;

public class Squar {
	int x = 0, y = 0;

	public Squar() {
		this.x = 0;
		this.y = 0;
	}

	public Squar(int X, int Y) {
		this.x = X;
		this.y = Y;
	}

	public int getX() {
		return x;
	}

	public void setX(int x) {
		this.x = x;
	}

	public int getY() {
		return y;
```

```
}

public void setY(int y) {
	this.y = y;
}


public Squar add(Squar SQ) {
	return new Squar(this.x + SQ.getX(), this.y + SQ.getY());
}

public Squar sub(Squar SQ) {
	return new Squar(this.getX() - SQ.getX(), this.getY() - SQ.getY());
}

public Squar turnR() {
	return new Squar(this.y, -this.x);
}

public Squar turnL() {
	return new Squar(-this.y, this.x);
}

public void display() {
	System.out.println("(" + this.x + " " + this.y + ")");
}
```

}

b. Code khối hình

```
package Class;

import java.awt.Color;
import java.util.Vector;

import GUI.ClassicJigsawPuzzle;

public class Cubes {
    int d[] = {0, 0, 0, 0, 0, 0, 0, 0};
    int c[] = {0, 1, 1, 1, 1, 1, 1, 0};
    int type = 0;
    int top, bot;
    int ic = 0;
    Squar tt = new Squar();
    Vector<Squar> v = new Vector();

    public Cubes() {
        this.tt = new Squar(2, 8);
        int rand = (int) (1000000 * Math.random() % 6 + 1);
        if (rand < 3) {
            if (Math.random() > 0.5)
                rand++;
        } else
            rand += 2;
        if (rand == 8)
            rand = 7;
        this.type = rand;
        this.ic = (int) (1000000 * Math.random() % 7 + 1);
        this.v = ininV();
        this.bot = c[type];
        this.top = d[type];
    }

    public int getIc() {
        return ic;
```

```
}

public void setIc(int ic) {
	this.ic = ic;
}

public Vector<Squar> ininV() {
	Vector<Squar> vii = new Vector();
	if (this.type == 1) {
		//tạo khối chữ T
		vii.add(new Squar(tt.getX(), tt.getY()));
		vii.add(new Squar(tt.getX(), tt.getY() + 1));
		vii.add(new Squar(tt.getX(), tt.getY() - 1));
		vii.add(new Squar(tt.getX() + 1, tt.getY() - 1));
	} else if (this.type == 2) {
		//tạo khối chữ L
		vii.add(new Squar(tt.getX(), tt.getY()));
		vii.add(new Squar(tt.getX(), tt.getY() + 1));
		vii.add(new Squar(tt.getX(), tt.getY() - 1));
		vii.add(new Squar(tt.getX() + 1, tt.getY() + 1));
	} else if (this.type == 3) {
		// tạo khối chữ S
		vii.add(new Squar(tt.getX(), tt.getY()));
		vii.add(new Squar(tt.getX(), tt.getY() - 1));
		vii.add(new Squar(tt.getX() + 1, tt.getY()));
		vii.add(new Squar(tt.getX() + 1, tt.getY() + 1));
	} else if (this.type == 4) {
		// tạo khối chữ Z
		vii.add(new Squar(tt.getX(), tt.getY()));
		vii.add(new Squar(tt.getX(), tt.getY() + 1));
		vii.add(new Squar(tt.getX() + 1, tt.getY()));
		vii.add(new Squar(tt.getX() + 1, tt.getY() - 1));
	} else if (this.type == 5) {
		// tạo khối
		vii.add(new Squar(tt.getX(), tt.getY()));
		vii.add(new Squar(tt.getX() + 1, tt.getY()));
		vii.add(new Squar(tt.getX(), tt.getY() + 1));
		vii.add(new Squar(tt.getX(), tt.getY() - 1));
	} else if (this.type == 6) {
		// tạo khối chữ O
```

```
                vii.add(new Squar(tt.getX(), tt.getY()));
                vii.add(new Squar(tt.getX(), tt.getY() + 1));
                vii.add(new Squar(tt.getX() + 1, tt.getY()));
                vii.add(new Squar(tt.getX() + 1, tt.getY() + 1));
        } else if (this.type == 7) {
                // tạo khối đg thẳng
                vii.add(new Squar(tt.getX(), tt.getY()));
                vii.add(new Squar(tt.getX(), tt.getY() - 1));
                vii.add(new Squar(tt.getX(), tt.getY() + 1));
                vii.add(new Squar(tt.getX(), tt.getY() + 2));
        }
        return vii;
}

public void turnRight() {
        if (type != 6) {
                Vector<Squar> tV = new Vector<>();
                for (int i = 0; i < this.v.size(); i++) {
                        Squar sq = this.v.elementAt(i);
                        sq = sq.sub(tt);
                        sq = sq.turnR();
                        sq = sq.add(tt);
                        tV.add(sq);
                }
                this.v = tV;
        }
}

public void turnLeft() {
        if (type != 6) {
                Vector<Squar> tV = new Vector<>();
                for (int i = 0; i < this.v.size(); i++) {
                        Squar sq = this.v.elementAt(i);
                        sq = sq.sub(tt);
                        sq = sq.turnL();
                        sq = sq.add(tt);
                        tV.add(sq);
                }
                this.v = tV;
        }
```

```
}

public void down() {
	this.tt = this.tt.add(new Squar(1, 0));
	Vector<Squar> tV = new Vector<>();
	for (int i = 0; i < this.v.size(); i++) {
		Squar sq = this.v.elementAt(i);
		sq = sq.add(new Squar(1, 0));
		tV.add(sq);
	}
	this.v = tV;
}

public void up() {
	this.tt = this.tt.sub(new Squar(1, 0));
	Vector<Squar> tV = new Vector<>();
	for (int i = 0; i < this.v.size(); i++) {
		Squar sq = this.v.elementAt(i);
		sq = sq.sub(new Squar(1, 0));
		tV.add(sq);
	}
	this.v = tV;
}

public void right() {
	this.tt = this.tt.add(new Squar(0, 1));
	Vector<Squar> tV = new Vector<>();
	for (int i = 0; i < this.v.size(); i++) {
		Squar sq = this.v.elementAt(i);
		sq = sq.add(new Squar(0, 1));
		tV.add(sq);
	}
	this.v = tV;
}

public void left() {
	this.tt = this.tt.sub(new Squar(0, 1));
	Vector<Squar> tV = new Vector<>();
	for (int i = 0; i < this.v.size(); i++) {
		Squar sq = this.v.elementAt(i);
```

```
                sq = sq.sub(new Squar(0, 1));
                tV.add(sq);
        }
        this.v = tV;
}

public boolean check(boolean b[][]) {
        for (int i = 0; i < this.v.size(); i++) {
                Squar sq = this.v.elementAt(i);
                if (b[sq.getX()][sq.getY()] == false) {
                        return false;
                }

        }
        return true;
}

public void display() {
        for (int i = 0; i < this.v.size(); i++) {
                Squar sq = this.v.elementAt(i);
                System.out.print("(" + (sq.getX()) + ", " + (sq.getY()) + ") ");
        }
        System.out.println("(" + tt.getX() + ", " + tt.getY() + ") ");
}

public Cubes(Squar TT) {
        this.tt = TT;
}

public int getType() {
        return type;
}

public void setType(int type) {
        this.type = type;
}

public Squar getTt() {
        return tt;
}
```

```
    public void setTt(Squar tt) {
        this.tt = tt;
    }

    public Vector<Squar> getV() {
        return v;
    }

    public void setV(Vector<Squar> v) {
        this.v = v;
    }

    public int getTop() {
        return top;
    }

    public void setTop(int top) {
        this.top = top;
    }

    public int getBot() {
        return bot;
    }

    public void setBot(int bot) {
        this.bot = bot;
    }

    public int initIc(int N) {
        int k = 0;
        do {
            k = (int) (1000000 * Math.random() % 6 + 1);
        } while (k == N);
        return k;
    }

}
```

c. Code giao diện

```
package GUI;

import java.awt.Color;
import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.*;
import java.awt.event.*;
import java.io.*;

import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.UnsupportedAudioFileException;
import javax.swing.*;

import Class.Cubes;
import Class.Squar;

import java.net.*;
import java.util.Scanner;
import java.util.Vector;
import java.io.*;

public class ClassicJigsawPuzzle extends JFrame implements KeyListener{
    Container cn;
    JPanel pn;
    Timer timer = new Timer(500, null);
    int M = 22, N = 10;
    int delay = 60;
    int index = delay;
```

```
    int score = 0;
    int line = 0;
    boolean die = false;
    JButton bt[][] = new JButton[M + 5][N + 9];
    boolean b[][] = new boolean[M + 5][N + 9];
    int preCl = 0;
    Color cl[] = {Color.black, Color.blue, Color.cyan, Color.green,
Color.magenta, Color.orange, Color.red, Color.yellow};
    Cubes p = new Cubes();
    Cubes [] Que = new Cubes[5];
    public ClassicJigsawPuzzle() {
        super("Tetris");
        cn = init();
        timer = new Timer(1, new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                if (index == 0) {
                    p.down();
                    if (!p.check(b))
                        newPuzz();
                    else {
                        try {
                            sound(4);
                        } catch
(UnsupportedAudioFileException | IOException | LineUnavailableException
e1) {
                            // TODO Auto-generated catch
block
                            e1.printStackTrace();
                        }
                    }
                    update();
                    index = delay;
                } else
                    index --;
            }
        });
        timer.start();
    }

    public Container init() {
        Container cn = this.getContentPane();

        pn = new JPanel();
```

```
        pn.setLayout(new GridLayout(M, N + 6));

        for (int i = 0; i < Que.length; i++) {
            Que[i] = new Cubes();
            Que[i].setIc(Que[i].initIc(preCl));
            preCl = Que[i].getIc();
        }

        for (int i = 0; i < M + 5; i++)
            for (int j = 0; j < N + 7; j++) {
                b[i][j]  = false;
            }

        for (int i = 0; i <= M + 3; i++)
            for (int j = 0; j <= N + 8; j++) {
                bt[i][j] = new JButton();
                bt[i][j].addKeyListener(this);
                bt[i][j].setBorder(BorderFactory.createMatteBorder(1,
1, 1, 1, Color.gray));
            }
        for (int i = 0; i <= M + 3; i++)
            for (int j = N + 3; j <= N + 8; j++) {
                bt[i][j].setBorder(null);
            }

        for (int i = 0; i < M + 3; i++)
            for (int j = 3; j < N + 3; j++) {
                b[i][j] = true;
            }

        for (int i = 3; i < M + 3; i++)
            for (int j = 3; j < N + 9; j++) {
                bt[i][j].setBackground(cl[0]);
                pn.add(bt[i][j]);
            }
        cn.add(pn);
        updateQue();
        this.setVisible(true);
        this.setSize(500, 700);
        this.setLocationRelativeTo(null);
        setResizable(false);
        setTextScoreLineSpeed();
```

```
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        return cn;
    }

    public void sound(int index) throws UnsupportedAudioFileException,
IOException, LineUnavailableException {
        try {
            File file = new File("Sound/" + index + ".wav");
            AudioInputStream audioStream =
AudioSystem.getAudioInputStream(file);
            Clip clip = AudioSystem.getClip();
            clip.open(audioStream);
            String response = "";
            clip.start();
        } catch (UnsupportedAudioFileException | IOException |
LineUnavailableException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }

    public int sub(int N) {
        int a[] = {1,   1,          5,          10,         0};
        int b[] = {0,   1,          0,          1,          0};
        int t = 0;
        for (int i = 0; i < a.length; i++) {
            t += a[i];
            if (N <= t)
                return b[i];
        }

        return 0;
    }

    public void setTextScoreLineSpeed() {
        int temp = 33;
        for (int j = N + 7; j >= N + 4; j--) {
            bt[M][j].setIcon(getIcon("" + temp));
            temp--;
        }
        temp = 22;
        for (int j = N + 6; j >= N + 5; j--) {
            bt[M - 2][j].setIcon(getIcon("" + temp));
```

```
                temp--;
        }

        temp = 13;
        for (int j = N + 7; j >= N + 4; j--) {
                bt[M - 4][j].setIcon(getIcon("" + temp));
                temp--;
        }
}


public void updateScore() {
        int temp = score;
        for (int j = N + 7; j >= N + 4; j--) {
                int k = temp % 10;
                temp /= 10;
                bt[M + 1][j].setIcon(getIcon("" + k));
        }

        temp = line;
        for (int j = N + 7; j >= N + 4; j--) {
                int k = temp % 10;
                temp /= 10;
                bt[M - 1][j].setIcon(getIcon("" + k));
        }

        temp = 61 - delay;
        for (int j = N + 7; j >= N + 4; j--) {
                int k = temp % 10;
                temp /= 10;
                bt[M - 3][j].setIcon(getIcon("" + k));
        }
}

public void updateQue() {
        for (int i = 0; i < M - 4; i++)
                for (int j = N + 3; j <= N + 8; j++) {
                        bt[i][j].setBackground(Color.black);
                        bt[i][j].setBorder(null);
                }

        p = Que[0];
        p.setTt(new Squar(2, 8));
```

```
            p.setV(p.ininV());
            if (!p.check(b)) {
                timer.stop();
                die = true;
            }
            for (int i = 0; i < Que.length - 1; i++)
                Que[i] = Que[i + 1];
            Que[Que.length - 1] = new Cubes();
            Que[Que.length - 1].setIc(Que[Que.length - 1].initIc(preCl));
            preCl = Que[Que.length - 1].getIc();
            int H = 3;
            for (int i = 0; i < Que.length; i++) {
                Que[i].setTt(new Squar(H + Que[i].getTop() + 1, N + 5));
                Que[i].setV(Que[i].ininV());
                for (int j = 0; j < Que[i].getV().size(); j++) {
                    Squar sq = Que[i].getV().elementAt(j);

    bt[sq.getX()][sq.getY()].setBackground(cl[Que[i].getIc()]);

    bt[sq.getX()][sq.getY()].setBorder(BorderFactory.createMatteBorder(1, 1,
1, 1, Color.gray));
                }
                H += Que[i].getBot() + Que[i].getTop() + 2;
            }
            updateScore();
    }

    public void update() {
            if (die){
                JOptionPane.showMessageDialog(null, "Your Score: " +
score);
                System.exit(0);
            }
            int countR = 0;
            for (int i = M + 2; i >= 1; i--) {
                boolean kt = true;
                for (int j = 3; j < N + 3; j++)
                    if (b[i][j] == true)
                        kt = false;
                if (kt) {
                    try {
                        sound(2);
                    } catch (UnsupportedAudioFileException |
```

```
IOException | LineUnavailableException e1) {
                                        // TODO Auto-generated catch block
                                        e1.printStackTrace();
                                }
                                countR++;
                                line++;
                                delay -= sub(line);
                                if (delay < 5)
                                        delay = 5;
                                updateScore();
                                for (int h = i; h >= 1; h--)
                                        for (int j = 3; j < N + 3; j++) {
                                                b[h][j] = b[h - 1][j];
                                                bt[h][j].setBackground(bt[h -
1][j].getBackground());
                                        }
                                for (int j = 3; j < N + 3; j++)
                                        b[0][j] = true;
                                i++;
                        }
                }

                switch (countR) {
                        case 1:
                                score += 1;
                                break;
                        case 2:
                                score += 4;
                                break;
                        case 3:
                                score += 7;
                                break;
                        case 4:
                                score += 10;
                                break;
                }
                updateScore();

                for (int i = 3; i < M + 3; i++)
                        for (int j = 3; j < N + 3; j++)
                                if (b[i][j])
                                        bt[i][j].setBackground(cl[0]);
```

```
            Vector<Squar> vP = p.getV();
            for (int i = 0; i < vP.size(); i++) {
                Squar sq = vP.elementAt(i);
                bt[sq.getX()][sq.getY()].setBackground(cl[p.getIc()]);
            }
    }

    public Icon getIcon(String index) {
            int w = 30;
            int h = 30;
            Image image = new ImageIcon(getClass().getResource("/Icons/" +
index + ".png")).getImage();
            Icon ic = new ImageIcon(image.getScaledInstance(w, h,
image.SCALE_SMOOTH));
            return ic;
    }

    @Override
    public void keyTyped(KeyEvent e) {
            // TODO Auto-generated method stub

    }

    public void newPuzz() {
            try {
                sound(2);
            } catch (UnsupportedAudioFileException | IOException |
LineUnavailableException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
            p.up();
            Vector<Squar> pp = p.getV();
            for (int i = 0; i < pp.size(); i++) {
                Squar sq = pp.elementAt(i);
                b[sq.getX()][sq.getY()] = false;
            }
            updateQue();
    }

    @Override
    public void keyPressed(KeyEvent e) {
            // TODO Auto-generated method stub
```

```
		if (die)
			return;
		if (e.getKeyCode() == e.VK_UP) {
			p.turnRight();
			if (!p.check(b)) {
				p.turnLeft();
			} else {
				try {
					sound(2);
				} catch (UnsupportedAudioFileException |
IOException | LineUnavailableException e1) {
					// TODO Auto-generated catch block
					e1.printStackTrace();
				}
			}
			update();
		} else if (e.getKeyCode() == e.VK_DOWN) {
			p.down();
			if (!p.check(b)) {
				newPuzz();
			} else {
				try {
					sound(1);
				} catch (UnsupportedAudioFileException |
IOException | LineUnavailableException e1) {
					// TODO Auto-generated catch block
					e1.printStackTrace();
				}
			}
			update();
		} else if (e.getKeyCode() == e.VK_LEFT) {
			p.left();
			if (!p.check(b)) {
				p.right();
			} else {
				try {
					sound(2);
				} catch (UnsupportedAudioFileException |
IOException | LineUnavailableException e1) {
					// TODO Auto-generated catch block
					e1.printStackTrace();
				}
			}
```

```
                update();
            } else if (e.getKeyCode() == e.VK_RIGHT) {
                p.right();
                if (!p.check(b)) {
                    p.left();
                } else {
                    try {
                        sound(2);
                    } catch (UnsupportedAudioFileException |
IOException | LineUnavailableException e1) {
                        // TODO Auto-generated catch block
                        e1.printStackTrace();
                    }
                }
                update();
            }
    }

    @Override
    public void keyReleased(KeyEvent e) {
        // TODO Auto-generated method stub

    }
}
```

d. Main

```
package Main;

import GUI.ClassicJigsawPuzzle;

public class Main {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		new ClassicJigsawPuzzle();
	}

}
```

IV. HƯỚNG PHÁT TRIỂN:

- Thêm vào các nút tương tác như nút pause, resume game
- Thêm các lựa chọn về độ khó để người chơi có thể tùy chọn
- Làm cho các hình khối khi rơi mượt mà hơn, di chuyển như rơi tự do
- Thêm 1 số loại hình khối